

listcompare

A file comparison tool

by Markus Gnam

© 2018

Program version: 2.0.0.1

Document version: 000004

Table of contents

USAGE	2
OPTIONS.....	2
Notes.....	3
Key files	3
Example 1.....	3
Value files.....	4
Example 2.....	4
Join files	4
Special cases for keys: Parts and concatenation, functions	4
More about functions	4
Special cases: Difference file.....	5
Example 3.....	5
Special cases: Conditions	5
Special cases: UTF-8.....	6
Special cases: --csv1	6
Special cases: Large file support	6
Debug files	6
Output files	6
Version history.....	7

listcompare.exe - Compare two lists based on a key

listcompare.exe - Compare two lists based on a key
(c) Markus Gnam 2018. **Version 2.0.0.1** 20180121

USAGE: listcompare [options] <file1> <file2>

Compare two files based on a key given for each file.

They don't need to be sorted. There is no size limit.

Defaults, if no options are specified:

Key for comparisons for both files is the first field.

Default field separator is white space (spaces/tabs).

Result output (if keys for these files exist):

key in both files => File: file1_and_file2.txt

key only in file1 => File: only_in_file1.txt

key only in file2 => File: only_in_file2.txt

For additional output of the value files use "-v"

Value files contain the whole line where the key occurs.

For all options and more details type "listcompare --help"

OPTIONS: Details about all options and their defaults:

Mandatory arguments to long options are mandatory for short options too.

- f, --fs=FIELDSEP The field separator [default: " "]
E.g. tabulator: --fs=\t or one space: --fs="[]"
- fs1=FIELDSEP The field separator of file 1 [default: " "]
- fs2=FIELDSEP The field separator of file 2 [default: " "]

- k, --key=FIELDNO The field number to compare [default: 1]
Use --key=0 to get the whole line as key.
For Special cases see Notes. E.g. part of field 8:
Field No[,Substr Pos][,Substr Length]: --key=8,1,4
A field name instead of a field number can be used:
E.g. --key=7 or with fname=: --key=fname=last_name
- key1=FIELDNO The field number of file 1 to compare [default: 1]
- key2=FIELDNO The field number of file 2 to compare [default: 1]

- condition1=RE Condition for key1 to be fulfilled [default: 0]
RE=<Field No>::~<regex in double quotes>. See Notes.
- condition2=RE Condition for key2 to be fulfilled [default: 0]
RE=<Field No>::~<regex in double quotes>. See Notes.

- v, --value={0,1} Generate files with values (whole lines)
additional to the key files [default: 0]

- j, --join={0,1} Generate "inner join" files [default: 0]

- d, --diff={0,1} Generate a difference file for values [default: 0]
--dif1=FIELDNO The field number of file 1 to compare [default: 0]
Use --dif1=0 for the whole line as value. See Notes.
--dif2=FIELDNO The field number of file 2 to compare [default: 0]
Use --dif2=0 for the whole line as value. See Notes.

- h, --header=N Take care of N header lines [default: 0]
--header1=N Take care of N header lines of file 1 [default: 0]
--header2=N Take care of N header lines of file 2 [default: 0]

```

--trim={0,1}      Trim the key [default: 1]
--ltrim={0,1}    Ltrim the key [default: 0]
--rtrim={0,1}    Rtrim the key [default: 0]

--upper={0,1}    Set the key to upper case [default: 0]
--lower={0,1}    Set the key to lower case [default: 0]
--lpadzero=N     Left pad the key with "0" to length N [default: 0]

--csv1={0,1}     Dequote CSV file for file 1 [default: 0]
--csv2={0,1}     Dequote CSV file for file 2 [default: 0]

--utf81={0,1}    For substring use with UTF-8 file 1 [default: 0]
--utf82={0,1}    For substring use with UTF-8 file 2 [default: 0]

--debug={0,1}    Print debug files [default: 0]
--largefiles=N   Split file into pieces of N lines [default: 1000000]
-?, --help      Display this help and exit

```

Notes:

You can use -, -- or / as first option character(s).
 E.g. -diff=1, --diff=1, /diff=1 are all valid alternatives.
 {0,1} means: Use "0" for No (False) or "1" for Yes (True).
 For these {0,1} options omitting these values means Yes.
 E.g. option -d, --diff={0,1}
 You can use either --diff=1 or simply --diff
 For the short option you can use -d, -d=1 or -d 1

Key files

contain the keys.
 Result output of the key files in sorted order (if keys exist):

```

key in both files => File: file1_and_file2.txt
key only in file1 => File: only_in_file1.txt
key only in file2 => File: only_in_file2.txt

```

E.g.

File numbers1.txt with content:

```

9
4

```

File numbers2.txt with content:

```

2
9

```

Example 1 (example1.cmd):

:: Most basic example: Compare keys of first field with FS white space:
listcompare numbers1.txt numbers2.txt

```

=> Result output of key files:
file1_and_file2.txt with content:
9
only_in_file1.txt with content:
4
only_in_file2.txt with content:
2

```

Further, key files should normally not contain any duplicates so they are written (if any) to key_file1_duplicates.txt and key_file2_duplicates.txt

Value files contain the whole line where the key occurs.

Result output of the value files in original order (if values exist):

```
key in both files => Values of file1: file1_and_file2_values_file1.txt
key in both files => Values of file2: file1_and_file2_values_file2.txt
key only in file1 => Values of file1: only_in_file1_values.txt
key only in file2 => Values of file2: only_in_file2_values.txt
```

Value files are interesting if the files contain more than one field.

Example 2 (example2.cmd) :

```
:: Field separator: Tab, Key: Field "last_name", values, one header line:
listcompare names1.txt names2.txt --fs=\t --key=fname=last_name -v -h=1
```

Join files behave exactly like a database inner join on a key for two tables. Note that the same field separator for each of the two files has to be used. Result file: joined_key_both_in_file1_and_file2_values.txt

E.g.

```
listcompare file1 file2 --fs=\t --header=1 --key=fname=city --join
```

Special cases for keys: Parts and concatenation, functions

```
Syntax: Field No[,Substr Pos][,Substr Length]
        [:Function name 1][,Param1],[,Param2] ...
        [:Function name 2][,Param1],[,Param2] ...
Repeat [+Field No of next field to concatenate] ... .. .
```

E.g. --key1=18,1,4:trim+7:leftpad,13,0

This means: From field 18 cut substring 1 to 4 and trim it.

Concatenate (+) with the result of:

Left pad field 7 with sign "0" to length 13.

E.g. --key2=17:trim+20:trim:leftpad,13,0

This means: Trim field 17. Concatenate (+) with the result of:

Trim field 20. Left pad this with sign "0" to length 13.

More about functions

Function arguments restrictions: No use of these signs: [:,+].

For use of a space sign put the whole key in double quotes.

The first argument has to be omitted when calling the function.

Available **functions** (only lower case for function names allowed):

leftpad(InputString,MyLength,MySign)

call: leftpad,MyLength,MySign

e.g.: --key1=20:leftpad,13,0

rightpad(InputString,MyLength,MySign)

call: rightpad,MyLength,MySign

e.g.: -key=2:rightpad,2,0

trim(InputString)

call: trim

e.g.: --key2=17:trim+20:trim:leftpad,13,0

ltrim(InputString)

call: ltrim

e.g.: --key1="18,1,4:ltrim:rtrim:suffix, +7:leftpad,13,0"

rtrim(InputString)

call: rtrim

e.g.: --key1=20,2,5+19,1,6:rtrim

```

prefix(InputString,MyPrefix)
  call: prefix,MyPrefix
  e.g.: --key2="17:ltrim:rtrim:prefix,co no +20:trim:leftpad,13,0"
suffix(InputString,MySuffix)
  call: suffix,MySuffix
  e.g.: --key=1:suffix,\t+2:prefix,\t
replace(InputString,from (*Global substitute regular expression*),to (*String*))
  call: replace,from,to
  e.g.: --key2=fname=GTIN:replace,.$,"":leftpad,12,0
sort(PartKeys)
  call: sort
  e.g.: --key1=1+2:sort

```

Special cases: Difference file

Option `--diff=1` to generate the file `differences_values.txt` (if results exist).

Difference file means different values for the same key. Default for the value is the whole line. This can be adjusted to a specific field number or further specifications for `dif1` and `dif2`. The "Special cases for keys: Parts and concatenation, functions" are valid for `dif1`, `dif2`, too. Especially the trim function is interesting.

```

E.g. ... --key=1 --diff --dif1=0:trim --dif2=0:trim
Or    ... --key=fname=GTIN --diff --dif1=fname=Title --dif2=fname=Title

```

Example 3 (example3.cmd):

```

:: Field separator: Tab, Key: Field 7, Difference file, one header line:
listcompare names1.txt names2.txt --fs=\t --key=7 --diff --header=1

```

```

=> Result file differences_values.txt with sample content:
*** Differences file1 and file2 for key: McLean
Tennessee TN Warren Mc Minnville 37110 Matt McLean
California CA San Diego San Diego 92126 Blake McLean

```

Hint: If the `--debug` option is used with the `--diff` option, additional files `differences_file1.txt` and `differences_file2.txt` are created with original content (whole line). They may be needed for database updates.

Special cases: Conditions

They are used for the corresponding key:

```

Syntax: Field No[,Substr Pos][,Substr Length]
        : (~|!~|==|!=|>|<|st)
        :<("regex" or "string")>
Repeat [[:Field No of next field to concatenate] ... ..]

```

```

E.g. --condition2=0::~:"^0102030405"
This means: For key2 the --condition2 value has to be fulfilled:
If the line ($0) doesn't start with 0102030405, key2 is skipped.
Field No: 0 => $0, 1 => $1 etc. The regex applies to this field.
Second parameter: operator. gt means greater than, st smaller than.
Third parameter: Regular expression or string in double quotes.

```

```

E.g. --condition2=0,1,10:=="0102030405"
This means: For key2 the --condition2 value has to be fulfilled:
If substring 1 to 10 isn't equal to 0102030405, key2 is skipped.

```

Special cases: UTF-8

Options `-utf81` for file 1 and `-utf82` for file 2:

These cases are rarely needed, only if you use substrings in keys and the substring contains multibyte characters with a UTF-8 encoded file, f.i. `--fs=\t --key1=19,4,5 --key2=fname="Column B",2,5 --utf81 --utf82`

Special cases: --csv1

for file1 dequoting and `--csv2` for file2 dequoting.

If a comma separated file is quoted according to csv rules, these options are needed. The key output is dequoted and the values output is original, f.i. `--fs=, --key1=7 --key2=9 --csv1=1 --csv2=1 --lpadzero=13`

Special cases: Large file support

This means: If the given line number limit (default: `--largefiles=1000000`) is reached, the file is split into pieces of temporary files with this number and is later merged together.

This applies only if needed. So there is no size limit for this program!

Debug files

These files are for debug purposes. They can be interesting,

e.g. `debug_key_file1.txt` shows the whole concatenated key of file1.

`debug_file1_unique_keys.txt` shows the whole line sorted by unique keys.

e.g. it is even possible to compare the same file using its name twice:

`listcompare test.csv test.csv -key1=1+2:sort -key2=1+2:sort -fs=, --debug`

`debug_file1_unique_keys.txt` shows unique results independent of the order of field 1 and field 2 by using the `:sort` function for concatenated keys.

`debug_file1_or_file2.txt` shows sorted union keys instead of intersection.

Output files

These output files will be created and deleted with program start:

Note that any existing files will be overwritten, without warning.

Key files (if results exist):

`file1_and_file2.txt`, `only_in_file1.txt`, `only_in_file2.txt`

`key_file1_duplicates.txt`, `key_file2_duplicates.txt`

Value files (if requested and if results exist):

`file1_and_file2_values_file1.txt`, `file1_and_file2_values_file2.txt`

`only_in_file1_values.txt`, `only_in_file2_values.txt`

Join files (if requested and if results exist):

`joined_key_both_in_file1_and_file2_values.txt`

Difference files (if requested and if results exist):

`differences_values.txt`

`differences_file1.txt`, `differences_file2.txt`

Debug files (if requested and if results exist):

`debug_key_file1.txt`, `debug_key_file2.txt`, `debug_file1_or_file2.txt`

`debug_key_file1_original.txt`, `debug_key_file2_original.txt`

`debug_key_file1_adjusted.txt`, `debug_key_file2_adjusted.txt`

`debug_key_file1_skipped.txt`, `debug_key_file2_skipped.txt`

`debug_file1_unique_keys.txt`, `debug_file2_unique_keys.txt`

Hint: These output files are created in the current directory (the directory from where `listcompare` is called). There should be only one call of `listcompare` in the same directory at the same time.

Version history

Version 2.0.0.1 20180121

- join option added. The new generation 2 of this software is able to behave exactly like a database inner join on a key for two tables.
- function replace added.

Version 1.0.1.3 20171004

- function rightpad added.
- function arguments check improved.
- Debug files differences_file1.txt and differences_file2.txt added.

Version 1.0.0.0 20170608

- Initial version.